Fundamentos de Algoritmia

Examen de enero Curso 2021/2022

Nombre: _____ **Grupo:** _____

Laboratorio: _____ **Puesto:** _____ **Usuario de DOMjudge:** _____

Normas de realización del examen

1. Debes programar soluciones para cada uno de los tres ejercicios, probarlas y entregarlas en el juez automático accesible en la dirección `http://exacrc`. Para la entrega el juez sólo tiene los datos de prueba del enunciado del problema.
2. Escribe comentarios que expliquen tu solución, justifiquen por qué se ha hecho así y ayuden a entenderla. Calcula la complejidad de todas las funciones que implementes.
3. En el juez te identificarás con el nombre de usuario y contraseña que has recibido al comienzo del examen. El nombre de usuario y contraseña que has estado utilizando durante la evaluación continua **no** son válidos.
4. Escribe tu **nombre y apellidos** en un comentario en la primera línea de cada fichero que subas al juez.
5. Tus soluciones serán evaluadas por el profesor independientemente del veredicto del juez automático. Para ello, el profesor tendrá en cuenta **exclusivamente** el último envío que hayas realizado de cada ejercicio.

Ejercicio 1

(2 puntos) En una empresa de distribución tienen establecidos tres tamaños de paquetes en función del peso del mismo: pequeños, medianos y grandes. Dada una lista de paquetes, se quieren reordenar de forma que se coloquen en primer lugar todos aquellos envíos con poco peso, a continuación todos los envíos con peso medio y por último todos los envíos de mucho peso.

Se pide:

1. Especifica una función que reciba un vector con el peso de cada pedido y modifique el vector para dejar al principio los paquetes con poco peso, a continuación los paquetes con peso medio y por último los paquetes con mucho peso. La función devolverá dos índices indicando dónde empiezan los paquetes medianos y dónde empiezan los paquetes grandes.

2. Implementa la función anterior. El coste de la implementación debe ser lineal en el número de paquetes.

3. Indica un invariante que permita probar la corrección del algoritmo implementado y justifica que el coste del algoritmo es lineal en el número de paquetes.

Entrada

La entrada consiste en una serie de casos de prueba. Cada caso de prueba consta de dos líneas: en la primera se indica el número n de paquetes, el peso a partir del cual se considera un paquete intermedio y el peso a partir del cual se considera un paquete grande; en la siguiente línea se da el peso de cada paquete.

El número de paquetes es mayor o igual que 0 y menor que 300.000. Los pesos son números enteros positivos que cumplen $0 < \text{peso} < 1000$. Y los pesos a partir de los cuales se consideran los paquetes mediano y grande cumplen $0 \leq \text{med} \leq \text{gran} \leq 1000$.

Salida

Para cada caso de prueba se escriben 3 líneas. En la primera línea se indican los pesos de los paquetes pequeños, en la siguiente línea los pesos de los paquetes medianos y en la última línea los pesos de los paquetes grandes. Si no existe ningún paquete de un tipo se dejará la línea vacía. Cada caso termina con una línea con tres guiones.

Entrada de ejemplo

```
6 6 8
4 5 6 7 8 9
5 50 70
10 20 30 80 90
3 50 50
50 50 50
```

Salida de ejemplo

```
4 5
6 7
8 9
---
10 20 30

80 90
---


50 50 50
---
```

Ejercicio 2

(2 puntos) Dado un vector de n elementos queremos saber si alguno de ellos aparece un número de veces estrictamente mayor que $n/2$. El problema debe resolverse utilizando la técnica de divide y vencerás en un tiempo del orden exacto de $n \log n$. Explica el algoritmo empleado, impleméntalo y justifica su coste.

Entrada

La entrada comienza con una línea que contiene el número de casos de prueba. Cada caso de prueba consta de una línea con los valores del vector, todos ellos enteros positivos. Cada línea acaba con el valor 0 que no pertenece al vector.

Salida

Para cada caso de prueba el programa escribirá en una línea el valor del elemento mayoritario si éste existe y `NO` si no existe ningún elemento mayoritario.

Entrada de ejemplo

```
9
7 2 2 2 4 2 7 2 0
5 5 5 5 5 0
4 4 4 8 8 8 2 2 2 0
0
3 0
3 4 0
3 4 3 0
3 4 3 5 0
3 4 3 5 3 6 0
```

Salida de ejemplo

```
2
5
NO
NO
3
NO
3
NO
NO
```

Ejercicio 3

(3 puntos) Este año me he propuesto salir a pasear todos los días. Desde mi casa salen n rutas y lo que tengo que decidir es cuál de ellas haré cada día. Voy a realizar una planificación de x días y después repetiré esa planificación durante todo el año. En esos x días no quiero repetir ninguna de las rutas; más aún, algunas rutas coinciden en algunos tramos y quiero evitar repetir un mismo tramo hasta que transcurran al menos dos días (un tramo visitado no se puede revisitar hasta al menos tres días más tarde). Para ello cuento con información sobre las rutas que coinciden en algún tramo. También voy a seleccionar aquellas rutas que cumpliendo los requisitos anteriores me gusten más, es decir, que la suma de lo que me gusta cada una de las rutas elegidas sea máxima.

1. Define el *espacio de soluciones* e indica cómo es el *árbol de exploración*.
2. Implementa un algoritmo de *vuelta atrás* que resuelva el problema. Explica claramente los *marcadores* que has utilizado.
3. Plantea una función de *poda de optimalidad* e impleméntala en tu algoritmo

Entrada

La entrada comienza con una línea que contiene el número de casos de prueba. Cada caso de prueba contendrá inicialmente el número de rutas n $(0 < n < 20)$ y el número de días que quiero planificar x $(0 < x \leq n \leq 10)$. A continuación, en una matriz simétrica de dimensión n por n se indica con un 1 si dos rutas coinciden en algún tramo y con un 0 si no coinciden. Por último, en una línea con n valores enteros positivos se indica lo que me gusta cada ruta: los números más altos indican que la ruta me gusta más.

Salida

Para cada caso de prueba, si se ha encontrado una lista de rutas que cumpla las restricciones el programa escribirá lo que me gustan las rutas seleccionadas y a continuación, separados por blancos, los números de cada ruta. En caso contrario se escribirá `IMPOSIBLE`. Si dos listas de rutas me gustan lo mismo elegiré la que tenga la ruta con menor número en primer lugar; si es el mismo, me quedaré con la de menor valor en la segunda posición, etc.

Entrada de ejemplo

```
4
4 3
1 0 0 0
0 1 1 0
0 1 1 0
0 0 0 1
1 9 9 6
5 4
1 0 0 0 0
0 1 1 0 1
0 1 1 0 0
0 0 0 1 0
0 1 0 0 1
1 9 8 6 4
5 4
1 0 0 0 0
0 1 1 0 1
0 1 1 0 0
0 0 0 1 0
0 1 0 0 1
1 9 9 6 4
2 2
1 1
1 1
3 3
```

Salida de ejemplo

```
16 0 1 3
24 1 0 3 2
25 1 0 3 2
IMPOSIBLE
```